**Assignment 4**

Data Visualization and Pre-processing

**Load the dataset.**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** statsmodels.api **as** sm

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.model\_selection **import** RandomizedSearchCV

**from** sklearn.metrics **import** mean\_squared\_error

**import** warnings

warnings**.**filterwarnings('ignore')

sns**.**set()

*# ----------------------- Helper functions -------------------*

**def** rmse(y\_true, y\_pred):

**return** np**.**sqrt(mean\_squared\_error(y\_true, y\_pred))

**def** eval\_model(model, X\_train, y\_train, X\_test, y\_test):

\_ **=** model**.**fit(X\_train, y\_train)

print("Train rmse : ", rmse(y\_train, model**.**predict(X\_train)))

print("Test rmse : ", rmse(y\_test, model**.**predict(X\_test)))

df **=** pd**.**read\_csv('C:/Users/Asus/Downloads/abalone.csv')

df

df['age'] **=** df**.**Rings **+** 1.5

*# remove rings variable*

df**.**drop('Rings', axis**=**1, inplace**=True**)

print("Data loaded Successfully!")

Data loaded Successfully!

**Univariate Analysis**

1. Summary Statistics

df['Sex']**.**value\_counts()

M 1528

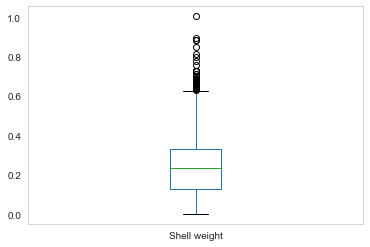
I 1342

F 1307

Name: Sex, dtype: int64

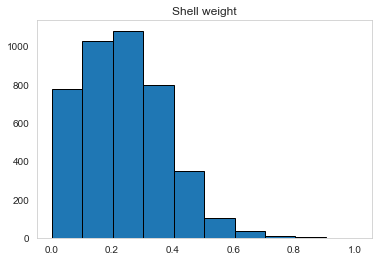
**Create charts**

df**.**boxplot(column**=**['Shell weight'], grid**=False**)

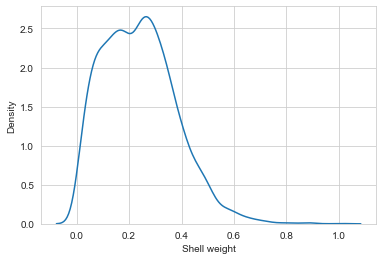


df**.**hist(column**=**'Shell weight', grid**=False**, edgecolor**=**'black')

array([[]], dtype=object)

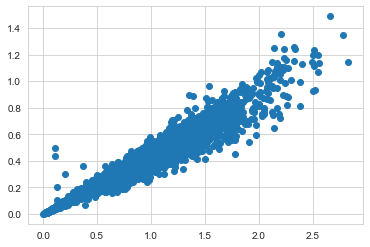


sns**.**kdeplot(df['Shell weight'])



**Bi - Variate Analysis**

plt**.**scatter(df['Whole weight'],df['Shucked weight'])



df**.**corr()

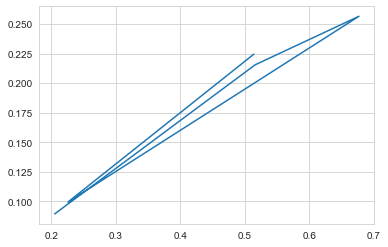
|  | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Length** | 1.000000 | 0.986812 | 0.827554 | 0.925261 | 0.897914 | 0.903018 | 0.897706 | 0.556720 |
| **Diameter** | 0.986812 | 1.000000 | 0.833684 | 0.925452 | 0.893162 | 0.899724 | 0.905330 | 0.574660 |
| **Height** | 0.827554 | 0.833684 | 1.000000 | 0.819221 | 0.774972 | 0.798319 | 0.817338 | 0.557467 |
| **Whole weight** | 0.925261 | 0.925452 | 0.819221 | 1.000000 | 0.969405 | 0.966375 | 0.955355 | 0.540390 |
| **Shucked weight** | 0.897914 | 0.893162 | 0.774972 | 0.969405 | 1.000000 | 0.931961 | 0.882617 | 0.420884 |
| **Viscera weight** | 0.903018 | 0.899724 | 0.798319 | 0.966375 | 0.931961 | 1.000000 | 0.907656 | 0.503819 |
| **Shell weight** | 0.897706 | 0.905330 | 0.817338 | 0.955355 | 0.882617 | 0.907656 | 1.000000 | 0.627574 |
| **age** | 0.556720 | 0.574660 | 0.557467 | 0.540390 | 0.420884 | 0.503819 | 0.627574 | 1.000000 |

plt**.**plot(df['Whole weight']**.**head() ,df['Shucked weight']**.**head(), )

plt**.**title('Line plot')

plt**.**xlabel('Whole weight')

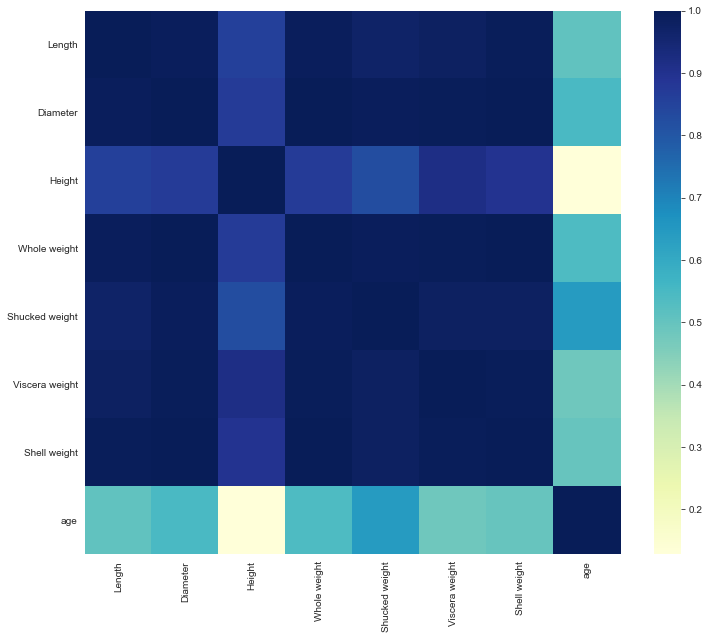
plt**.**ylabel('Shucked weight')



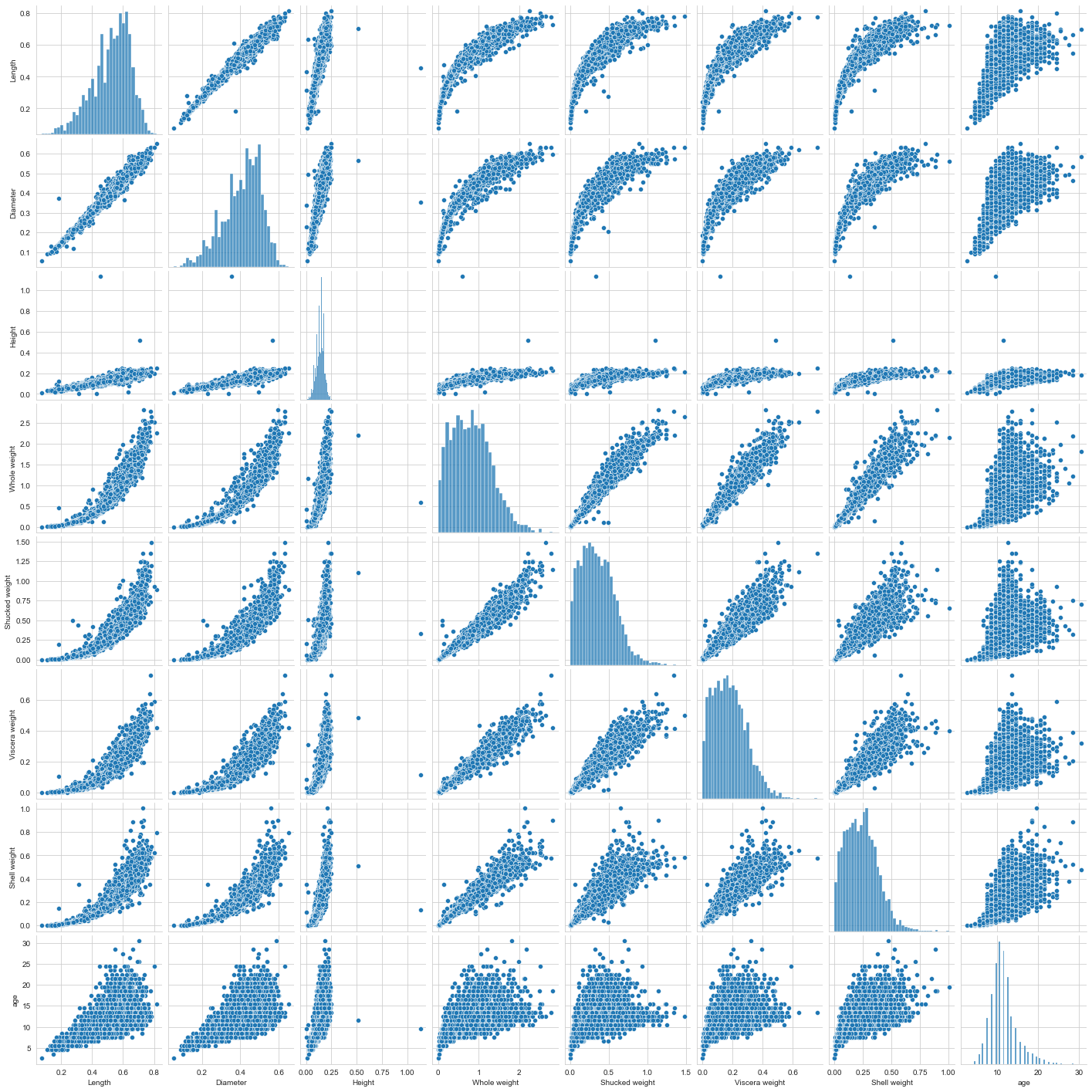
**Multivariate analysis**

f **=** plt**.**subplots(figsize**=**(12,10))

sns**.**heatmap(df**.**head()**.**corr(), cmap**=**"YlGnBu")



sns**.**pairplot(df)

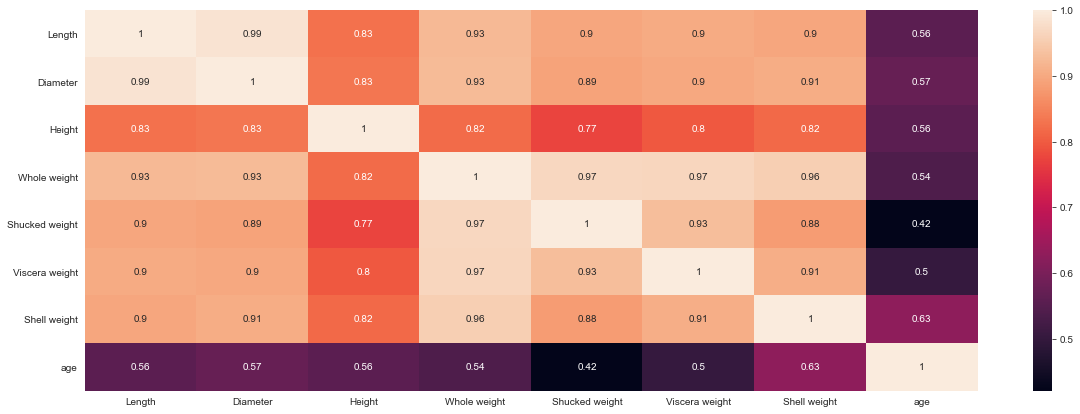


numerical\_features **=** df**.**select\_dtypes(include **=** [np**.**number])**.**columns

categorical\_features **=** df**.**select\_dtypes(include **=** [np**.**object])**.**columns

plt**.**figure(figsize **=** (20,7))

sns**.**heatmap(df[numerical\_features]**.**corr(),annot **=** **True**)



**Perform descriptive statistics on the dataset.**

df**.**shape

(4177, 9)

df**.**columns

Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',

'Viscera weight', 'Shell weight', 'age'],

dtype='object')

df**.**dtypes

Sex object

Length float64

Diameter float64

Height float64

Whole weight float64

Shucked weight float64

Viscera weight float64

Shell weight float64

age float64

dtype: object

df**.**info()

RangeIndex: 4177 entries, 0 to 4176

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Sex 4177 non-null object

1 Length 4177 non-null float64

2 Diameter 4177 non-null float64

3 Height 4177 non-null float64

4 Whole weight 4177 non-null float64

5 Shucked weight 4177 non-null float64

6 Viscera weight 4177 non-null float64

7 Shell weight 4177 non-null float64

8 age 4177 non-null float64

dtypes: float64(8), object(1)

memory usage: 293.8+ KB

df**.**describe()

df**.**head()

|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 16.5 |
| **1** | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 8.5 |
| **2** | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 10.5 |
| **3** | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 11.5 |
| **4** | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 8.5 |

df**.**tail()

|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4172** | F | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 12.5 |
| **4173** | M | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 11.5 |
| **4174** | M | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 10.5 |
| **4175** | F | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 11.5 |
| **4176** | M | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 13.5 |

|  | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 |
| **mean** | 0.523992 | 0.407881 | 0.139516 | 0.828742 | 0.359367 | 0.180594 | 0.238831 | 11.433684 |
| **std** | 0.120093 | 0.099240 | 0.041827 | 0.490389 | 0.221963 | 0.109614 | 0.139203 | 3.224169 |
| **min** | 0.075000 | 0.055000 | 0.000000 | 0.002000 | 0.001000 | 0.000500 | 0.001500 | 2.500000 |
| **25%** | 0.450000 | 0.350000 | 0.115000 | 0.441500 | 0.186000 | 0.093500 | 0.130000 | 9.500000 |
| **50%** | 0.545000 | 0.425000 | 0.140000 | 0.799500 | 0.336000 | 0.171000 | 0.234000 | 10.500000 |
| **75%** | 0.615000 | 0.480000 | 0.165000 | 1.153000 | 0.502000 | 0.253000 | 0.329000 | 12.500000 |
| **max** | 0.815000 | 0.650000 | 1.130000 | 2.825500 | 1.488000 | 0.760000 | 1.005000 | 30.500000 |
| df**.mode()** |  |  |  |  |  |  |  |  |

df**.**mode()

| **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | 0.550 | 0.45 | 0.15 | 0.2225 | 0.175 | 0.1715 | 0.275 | 10.5 |
| NaN | 0.625 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
|  |  |  |  |  |  |  |  |  |

**Handle the Missing values.**

print(df**.**isnull()

Sex Length Diameter Height Whole weight Shucked weight \

0 False False False False False False

1 False False False False False False

2 False False False False False False

3 False False False False False False

4 False False False False False False

... ... ... ... ... ... ...

4172 False False False False False False

4173 False False False False False False

4174 False False False False False False

4175 False False False False False False

4176 False False False False False False

Viscera weight Shell weight age

0 False False False

1 False False False

2 False False False

3 False False False

4 False False False

... ... ... ...

4172 False False False

4173 False False False

4174 False False False

4175 False False False

4176 False False False

[4177 rows x 9 columns]

df**.**isna()**.**sum()

Sex 0

Length 0

Diameter 0

Height 0

Whole weight 0

Shucked weight 0

Viscera weight 0

Shell weight 0

age 0

dtype: int64

**Find the outliers and replace the outliers**

In [159]:

train, test **=** train\_test\_split(df, test\_size**=**0.25, random\_state**=**1)

print('Train data points :', len(train))

print('Test data points :', len(test))

Train data points : 3132

Test data points : 1045

**Variable separation**

numerical\_features **=** ["Length", 'Diameter', 'Height','Whole weight',

'Shucked weight', 'Viscera weight', 'Shell weight']

categorical\_feature **=** "Sex"

features **=** numerical\_features **+** [categorical\_feature]

target **=** 'age'

**Target distribution**

fig, axes **=** plt**.**subplots(ncols**=**2,figsize**=**(16, 5))

train[target]**.**plot**.**hist(color**=**'blue', ax**=**axes[0])

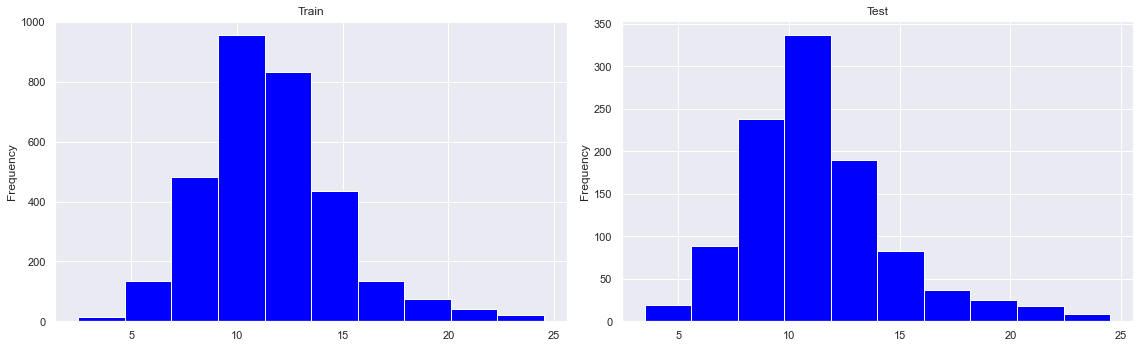
axes[0]**.**set(title**=**"Train")

test[target]**.**plot**.**hist(color**=**'blue', ax**=**axes[1])

axes[1]**.**set(title**=**"Test")

plt**.**tight\_layout()

plt**.**show()



**Distribution of numerical features**

In [203]:

fig, axes **=** plt**.**subplots(4,2,figsize**=**(16, 14))

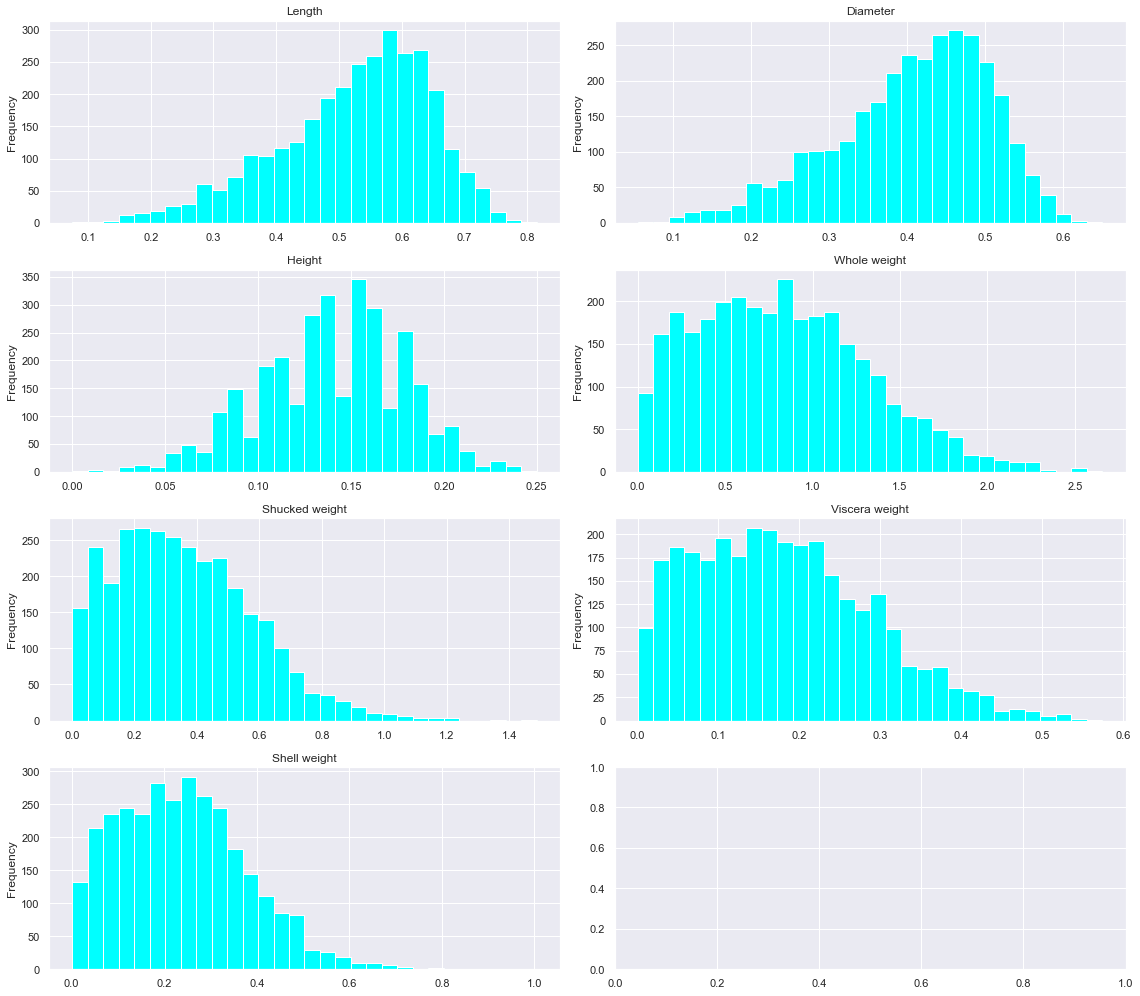
axes **=** np**.**ravel(axes)

**for** i, c **in** enumerate(numerical\_features):

hist **=** train[c]**.**plot(kind **=** 'hist', ax**=**axes[i], title**=**c, color**=**'cyan', bins**=**30)

plt**.**tight\_layout()

plt**.**show()



**Boxplot(Outliers)**

fig, axes **=** plt**.**subplots(4,2,figsize**=**(16, 14))

axes **=** np**.**ravel(axes)

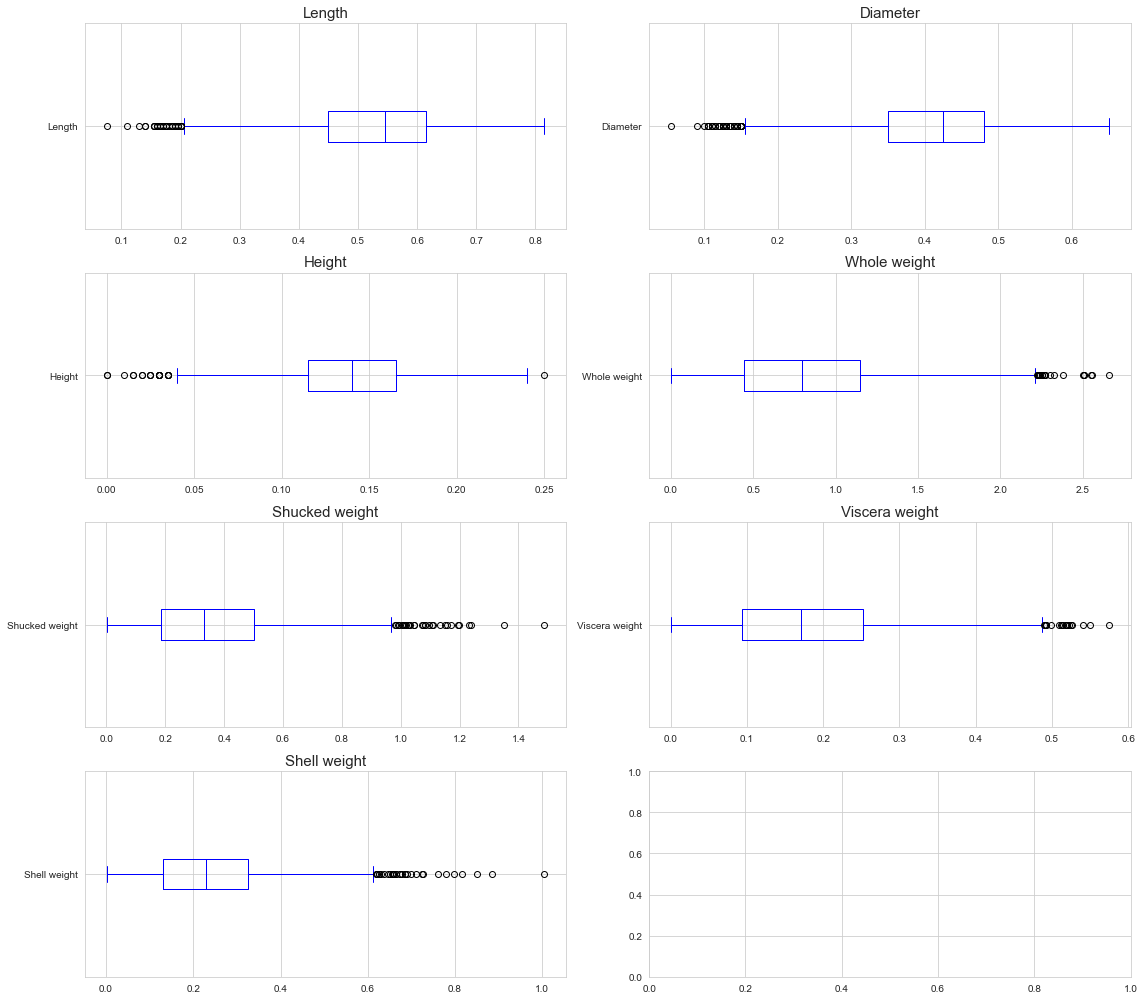
**for** i, c **in** enumerate(numerical\_features):

hist **=** train[c]**.**plot(kind **=** 'box', ax**=**axes[i],color**=**'blue', vert**=False**)

axes[i]**.**set\_title(c, fontsize**=**15)

plt**.**tight\_layout()

plt**.**show()



**Pie chart : Categorical feature sex**

t **=** train[categorical\_feature]**.**value\_counts(normalize**=True**)

t**.**plot(kind**=**'pie',

figsize**=**(5,5),

title**=**categorical\_feature,

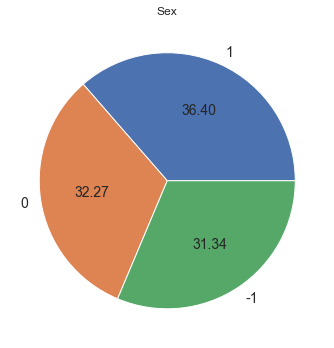
ylabel**=**"",

autopct**=**"%.2f",

fontsize**=**14)

plt**.**tight\_layout()

plt**.**show()



**Pearson Correlation**

plt**.**figure(figsize**=**(14,6))

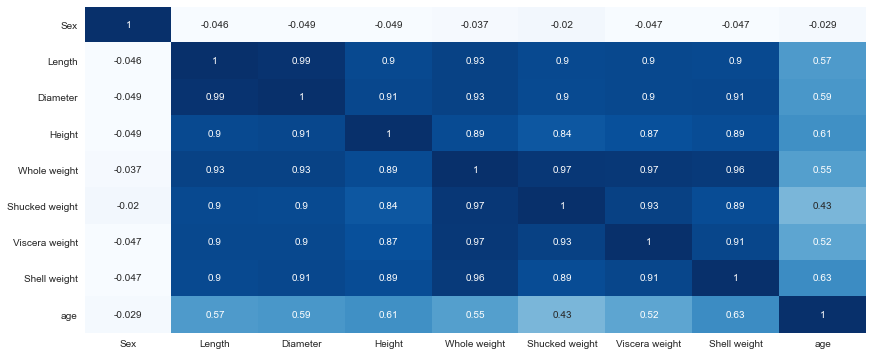
sns**.**heatmap(train**.**corr(method**=**'pearson'),

annot**=True**,

cbar**=False**,

cmap**=**'Blues')

plt**.**show()



**Height Vs Length Vs age**

fig **=** plt**.**figure(figsize**=**(10,8))

ax **=** plt**.**axes(projection**=**'3d')

ax**.**set\_xlabel('Height of abalone (mm)')

ax**.**set\_ylabel('Length of abalone (mm)')

ax**.**set\_zlabel('age')

ax**.**scatter3D(train['Height'],

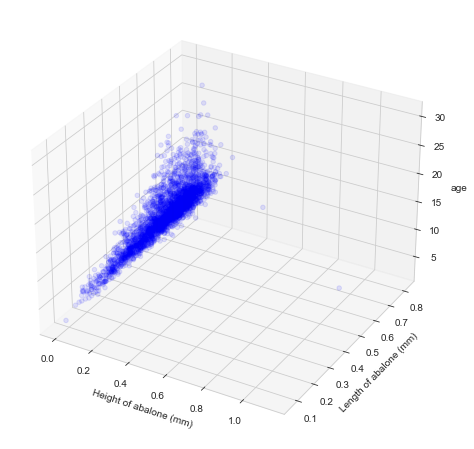
train['Length'],

train[target],

c**=**'blue',

alpha**=**0.1);

plt**.**show()



**Scatter plot**

fig, axes **=** plt**.**subplots(4,2,figsize**=**(16, 18))

axes **=** np**.**ravel(axes)

**for** i, c **in** enumerate(numerical\_features):

\_ **=** sns**.**scatterplot(x**=**train[c],

y**=**train[target],

ax**=**axes[i],

color**=**'blue')

axes[i]**.**set\_title(f"{c} Vs age",

fontsize**=**14,

fontweight**=**'bold')

axes[i]**.**set\_xlabel(c, fontsize**=**15)

axes[i]**.**set\_ylabel('age', fontsize**=**15)

plt**.**tight\_layout()

plt**.**show()



**Label Encoding**

train**.**Sex **=** train**.**Sex**.**replace({"M":1, "I":0, "F":**-**1})

test**.**Sex **=** test**.**Sex**.**replace({"M":1, "I":0, "F":**-**1})

**Removing outliers**

idx **=** train**.**loc[train**.**Height**>**0.4]**.**index

train**.**drop(idx, inplace**=True**)

idx **=** train**.**loc[train['Viscera weight']**>**0.6]**.**index

train**.**drop(idx, inplace**=True**)

idx **=** train**.**loc[train[target]**>**25]**.**index

train**.**drop(idx, inplace**=True**)

**Split the data into dependent and independent variables**

*# Splitting the Dataset into the Independent*

X **=** df**.**iloc[:, :**-**1]**.**values

print(X)

[['Length']

['Diameter']

['Height']

['Whole weight']

['Shucked weight']

['Viscera weight']

['Shell weight']

['Sex']]

*# Extracting the Dataset to Get the Dependent*

Y **=** df**.**iloc[:, **-**1]**.**values

print(Y)

[0.08050966929596948 0.10924659159919295 0.13527752763066053

0.15710514425282232 0.1203996256444543 0.1247626207310252

0.2479560616535119 0.024742759192363375]

**Scale the independent variables**

**from** sklearn.preprocessing **import** scale

x **=** scale(Y)

x

array([-0.74468165, -0.26368143, 0.17202583, 0.5373777 , -0.07700132,

-0.00397327, 2.05804545, -1.67811132])

**Split the data into training and testing**

train, test **=** train\_test\_split(df, test\_size**=**0.25, random\_state**=**1)

print('Train data points :', len(train))

print('Test data points :', len(test))

Train data points : 3132

Test data points : 1045

**Build the model**

**Feature seperation**

X\_train **=** train[features]

y\_train **=** train[target]

X\_test **=** test[features]

y\_test **=** test[target]

X\_train**.**head()

|  | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **Sex** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4014** | 0.625 | 0.480 | 0.175 | 1.0650 | 0.4865 | 0.2590 | 0.285 | 1 |
| **3252** | 0.480 | 0.380 | 0.130 | 0.6175 | 0.3000 | 0.1420 | 0.175 | 1 |
| **305** | 0.200 | 0.145 | 0.060 | 0.0370 | 0.0125 | 0.0095 | 0.011 | 0 |
| **1857** | 0.505 | 0.400 | 0.145 | 0.7045 | 0.3340 | 0.1425 | 0.207 | 0 |
| **439** | 0.500 | 0.415 | 0.165 | 0.6885 | 0.2490 | 0.1380 | 0.250 |  |

**Base models**

models **=** {'linear\_regression':LinearRegression(),

'decision\_tree':DecisionTreeRegressor(random\_state**=**1),

'random\_forest':RandomForestRegressor(random\_state**=**1),

}

**for** key, regressor **in** models**.**items():

print(key)

eval\_model(regressor, X\_train, y\_train, X\_test, y\_test)

print("\n------------------------------------------")

linear\_regression

Train rmse : 2.1601637766834694

Test rmse : 2.199332649510367

------------------------------------------

decision\_tree

Train rmse : 0.0

Test rmse : 2.8672378052018894

------------------------------------------

random\_forest

Train rmse : 0.7983734867135102

Test rmse : 2.1456051220373515

------------------------------------------

**Running the model**

*# Linear regression*

lr\_params **=** {'fit\_intercept':[**True**,**False**]}

*# Decision tree*

dt\_params **=** {'max\_depth': [4, 6, 8, 10, 12, 14, 16, 20],

'min\_samples\_split': [5, 10, 20, 30, 40, 50],

'max\_features': [0.2, 0.4, 0.6, 0.8, 1],

'max\_leaf\_nodes': [8, 16, 32, 64, 128,256]}

*# Random Forest*

rf\_params **=** {'bootstrap': [**True**, **False**],

'max\_depth': [2, 5, 10, 20, **None**],

'max\_features': ['auto', 'sqrt'],

'min\_samples\_leaf': [1, 2, 4],

'min\_samples\_split': [2, 5, 10],

'n\_estimators': [100, 150, 200, 250]}

params **=** [lr\_params, dt\_params, rf\_params, ]

*# searching Hyperparameters*

i**=**0

**for** name, model **in** models**.**items():

print(name)

regressor **=** RandomizedSearchCV(estimator **=** model,

n\_iter**=**10,

param\_distributions **=** params[i],

cv **=** 3,

scoring **=** 'neg\_root\_mean\_squared\_error')

search **=** regressor**.**fit(X\_train, y\_train)

print('Best params :',search**.**best\_params\_)

print("RMSE :", **-**search**.**best\_score\_)

i**+=**1

print()

linear\_regression

Best params : {'fit\_intercept': True}

RMSE : 2.1685158384088488

decision\_tree

Best params : {'min\_samples\_split': 30, 'max\_leaf\_nodes': 16, 'max\_features': 0.8, 'max\_depth': 14}

RMSE : 2.320260154702674

random\_forest

Best params : {'n\_estimators': 150, 'min\_samples\_split': 2, 'min\_samples\_leaf': 4, 'max\_features': 'auto', 'max\_depth': None, 'bootstrap': True}

RMSE : 2.1309509089096124

**Final modeling**

Random forest regressor is performing better Train the model

rf\_params **=** {'n\_estimators': 200,

'min\_samples\_split': 2,

'min\_samples\_leaf': 4,

'max\_features': 'sqrt',

'max\_depth': **None**,

'bootstrap': **True**}

model **=** RandomForestRegressor(random\_state**=**1, **\*\***rf\_params)

model**.**fit(X\_train, y\_train)

RandomForestRegressor(max\_features='sqrt', min\_samples\_leaf=4, n\_estimators=200,

random\_state=1)

**Saving the model**

**import** pickle

**with** open("model.pkl", "wb") **as** f:

pickle**.**dump(model, f)

**Evaluation**

print("Train rmse : ", rmse(y\_train, model**.**predict(X\_train)))

print("Test rmse : ", rmse(y\_test, model**.**predict(X\_test)))

Train rmse : 1.5313840467501842

Test rmse : 2.146490954202156

**Feature importance**

df **=** pd**.**DataFrame([features, model**.**feature\_importances\_])**.**T

df**.**columns **=** ['feature', 'importance']

df**.**sort\_values("importance", ascending**=False**)

df**.**sort\_values("importance", ascending**=False**)

|  | **feature** | **importance** |
| --- | --- | --- |
| **6** | Shell weight | 0.247956 |
| **3** | Whole weight | 0.157105 |
| **2** | Height | 0.135278 |
| **5** | Viscera weight | 0.124763 |
| **4** | Shucked weight | 0.1204 |
| **1** | Diameter | 0.109247 |
| **0** | Length | 0.0805097 |
| **7** | Sex | 0.0247428 |

**Scatter plot**

y\_pred **=** model**.**predict(X\_test)

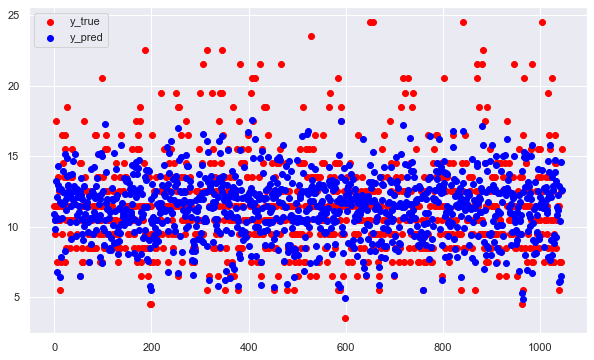
fig **=** plt**.**figure(figsize**=**(10, 6))

plt**.**scatter(range(y\_test**.**shape[0]), y\_test, color**=**'red', label**=**'y\_true')

plt**.**scatter(range(y\_test**.**shape[0]), y\_pred, color**=**'blue', label**=**'y\_pred')

plt**.**legend()

plt**.**show()

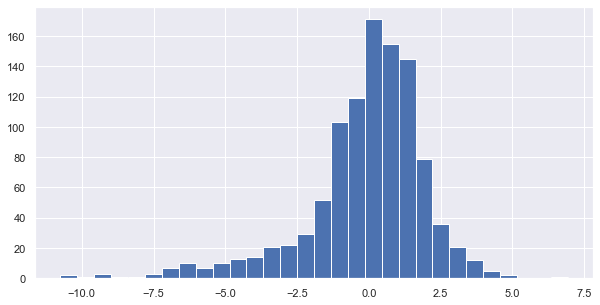


**Error distribution**

plt**.**figure(figsize**=**(10,5))

plt**.**hist(y\_pred**-**y\_test, bins**=**30)

plt**.**show(



**Test the model**

**def** predict\_age(x):

x **=** pd**.**DataFrame([x], columns**=**features)

age **=** model**.**predict(x)

**return** round(age[0],2)

**with** open("model.pkl", 'rb') **as** f:

model **=** pickle**.**load(f)

*# Random sample from test set*

ex **=** [0.295 , 0.225 , 0.08 , 0.124 , 0.0485, 0.032 , 0.04 , 0.]

print("Estimated age : ",predict\_age(ex))

Estimated age : 8.86

**Measure the performance using Metrics.**

y\_pred **=** regressor**.**predict(X\_test)

**from** sklearn **import** metrics

print('Mean Absolute Error:', metrics**.**mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics**.**mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np**.**sqrt(metrics**.**mean\_squared\_error(y\_test, y\_pred)))

Mean Absolute Error: 1.5098937086684512

Mean Squared Error: 4.61800632962725

Root Mean Squared Error: 2.1489547062763443